フィルタリングポリシをトラヒック特性に適応させる機能を有するパケットキャプチャシステム

中村　陸† 片山　喜章†† 高橋　直久††

† 南山大学大学院数理情報研究科数理情報専攻　〒 489–0863 愛知県瀬戸市せいれい町 27
† 名古屋工業大学大学院工学研究科情報工学専攻　〒 466–8555 名古屋市昭和区御器所町
E-mail: †riku@moss.elcom.nitech.ac.jp, ††{katayama,naohisa}@nitech.ac.jp

あらまし　我々は空間分割型パケット分類法を用いたパケットキャプチャシステムの開発を行っている. このシステムは 10Gbps 以上の高速なネットワークを流れるパケットをリアルタイムに収集するために, パケット分類器をオンチップで実現し, パソコンのネットワークインターフェースカードに搭載する. しかし, 空間分割型パケット分類法は膨大なメモリを要する場合がある. この問題の解決法として, フィルタを変化させ必要なメモリを抑える方法がある. しかし, フィルタを変化させると本来キャプチャする必要のないパケット（ノイズとよぶ）までキャプチャしてしまうことがある. そこで, 本稿ではノイズを低減するようにフィルタリングポリシを事前にキャプチャしたトラヒックの特性に対して適応的に変化させる機能を持つ空間分割型パケットキャプチャシステムを提案する. 提案システムは次の 2 つの機能を有する. (1) ノイズの特性を分析する機能, (2) 分析結果に従ってフィルタリングポリシをノイズが低減するように変更する機能. 評価実験により, フィルタリングポリシを事前にキャプチャしたトラヒック特性に対して適応的に変化させることにより, ノイズを低減させることが可能であることを確認した.

キーワード　パケットキャプチャ, フィルタリングポリシ, 近似手法, トラヒック特性, パケット分類器

A Packet Capture System Adjusting a Filtering Policy to Traffic Characteristics

Riku NAKAMURA†, Yoshiaki KATAYAMA††, and Naohisa TAKAHASHI††

† Graduate School of Mathematical Sciences and Information Engineering, Nanzan University　27 Seirei, Seto, Aichi, 489-0863 Japan
† Graduate School of Engineering, Nagoya Institute of Technology　Gokiso, Showa, Nagoya, 466-8555 Japan
E-mail: †riku@moss.elcom.nitech.ac.jp, ††{katayama,naohisa}@nitech.ac.jp

Abstract　We propose a high-speed packet capture system that reduces an amout of memories in a packet classifier by adjusting a filtering polcy to traffic characteristics. It has following two features. (1) It analyses the noise involved in the captured packets. (2) It modifies the capture policy so as to reduce the noise. The experimental results show that proposed system is effective for the reduction of the noise.

Key words　Packet Capture, Filtering Policy, Approximate Method, Traffic Characteristics, Packet Classifier

1. はじめに

ネットワークの安定運用には, 悪意のあるホストからの攻撃や, ネットワーク機器の動作を監視・診断することが不可欠である. 監視・診断の方法としてパケットを収集し, その中身から状態を把握するパケットキャプチャシステムが広く使われている.

しかし, 10Gbps 以上の高速なネットワークでは膨大な量のパケットが短時間で流れることがある. このような場合, パケットキャプチャシステム内部での転送速度やディスクの書き込み速度, 記憶装置の容量といったハードウェア上の制限や, キャプチャ後のパケット処理が追いつかないといったソフトウェア上の制限により, リアルタイムで流れるパケット全てを収集し, 監視・診断を行うのは非常に困難である.

この問題を解決するために, ネットワークインターフェースでフィルタリング技術 [1]〜 [7] を用いて監視・診断に必要なパケットだけを選択してキャプチャする方法が有効になる. この方法を用いればパケットキャプチャシステムが処理するトラヒック量を減らす事ができる. しかし, 高速なネットワークでパケッ

トのパケットフィルタリングを行うにはフィルタリング速度も高速でなければならない.

一方, 多数のフィルタに対して高速なフィルタリングが行える方式として空間分割型のパケット分類方式がある [2]〜[7]. 空間分割型のパケット分類方式ではパケットをそのヘッダの値を座標とする点とするパケット空間を作る．パケットキャプチャはフィルタの条件に一致したパケットをキャプチャする．この空間において, フィルタはキャプチャすべきパケットの点を含むフィルタ領域として表す．全てのフィルタ領域を表す分類表を作成し, これを用いることで高速にパケットの分類が可能となる．しかし, フィルタの数が増加すると, 分類表も大きくなり, 大容量のメモリが必要となる.

この問題の解決法としてフィルタリングポリシを変更して必要なメモリ量を減らす手法 [7] が提案されている. しかし, フィルタリングポリシを変更すると本来キャプチャする必要のないパケット (**ノイズ**と呼ぶ) までキャプチャされてしまうことがある. ノイズを低減するにはどのフィルタをどのように変更するかが重要となる. 変更するフィルタの選択法として, パケット空間においてフィルタの変化した領域に含まれる点の数が最も少なくなるフィルタを選択する手法 [7] がある. しかし, この方式ではパケットの分布に偏りがあるとノイズが上昇してしまう.

そこで本稿では, 以下の特徴を持つパケットキャプチャシステム提案する.

特徴 1 キャプチャしたパケットにからパケット空間の各点に対するパケット出現頻度を推定する機能を実現する. これにより, フィルタを変更することにより生じるノイズ量を推定することが可能となる.

特徴 2 パケット出現頻度に従ってフィルタリングポリシを分類表が小さくなるよう変更する機能を実現する. これにより, メモリを任意の容量以下に抑えつつ, 従来の方式と比べノイズの量を減らすことが可能となる.

本稿の構成は次の通り. まず第 2 章ではパケットキャプチャの高速化手法の関連研究について述べる. 第 3 章ではパケットキャプチャの高速化手法について述べ, 第 4 章で提案方式の実現法について述べ, 第 5 章では評価実験について述べる.

2. 関 連 研 究

2.1 高速ネットワークでのパケットキャプチャ手法

既存の研究において, ネットワークインターフェースでパケットに処理を行い, パケットキャプチャシステム内部で処理するパケットを減らす方法がいくつか提案されている.

サンプリングパケットを保存する手法として, Cisco 社の NetFlow [10] がある．NetFlow では, 受信したパケットの内, 一部のパケットをランダムにキャプチャ(サンプリング) することで, キャプチャするパケットを削減する．しかし, この手法では監視・診断に必要なパケットまで廃棄されてしまう可能性がある.

圧縮したパケットを保存する手法として, 清水奨らの手法 [9] がある．この手法では, パケットを複数個まとめてリアルタイムで圧縮することにより, パケットキャプチャシステム内部に流れるデータ量を削減する．文献 [9] によるとランダムなデータパターンを持つパケットを 50 しかし, 圧縮率を高くすると, 圧縮処理にかかる計算コストが増加してしまう問題がある.

これらの手法はそれぞれ単独で用いるだけでなく, 複数の手法を組み合わせて使用することが可能な場合がある．例えば, まずパケットをフィルタリングし, その後フィルタを通過したパケットを圧縮するといったような使い方である．このように複数の手法を組み合わせることにより, パケットキャプチャシステムをより高速なネットワークに対応させることができる.

本稿ではフィルタリング手法をのみ対象とし, 複数の手法を組み合わせて用いる場合には, 既存の手法を共に用いる.

2.2 パケット分類手法

従来のパケット分類ではパケットとその分類条件を順に比較し, 一致するフィルタを見つける．しかし, この手法は分類に必要なメモリは少量ですむが, 分類速度が遅いため, 高速なネットワークでは適用が難しい．そのため, パケットの分類の高速化手法が提案されている．パケット分類について, 高速ルーティングテーブル探索のために連想メモリ（CAM）を用いた手法 [1] がある．この方式では, 簡単なハードウェアで高速なパケットの分類が可能であるが, 高コストになるため分類条件が複雑な場合には適用が難しい.

ソフトウェアによる実現方式として, V.Srinvasan らの手法 [2] や Pankaj Gupta らの手法 [4] がある．これらの手法ではどちらも次元毎に分割して積を求める空間分割によるパケットの分類を行っている．文献 [4] の手法では考えられるパケット全ての分類結果を事前にソフトウェアで計算するので, その結果を記述した表を格納するメモリと, それを参照するだけの安価なハードウェアで実現できる．しかし, フィルタの数やキーの数が多くなると, 多くのメモリが必要になる．近年の高速なネットワークに対応するためには表をきわめて高速なメモリ（プロセッサ内部のキャッシュメモリ等）に収める必要があり, このようなメモリを多く用意するときわめて高価になってしまう．本稿では本稿ではこれらの手法の問題点である, 多くのメモリが必要という問題点を解決する.

3. 空間分割型パケット分類器

3.1 機 能 概 要

パケット分類器では, パケットのヘッダの値に従い, IP パケットを識別する [8]. ここで, ヘッダは送信先 IP アドレス, 送信先ポート番号など, それぞれ長さ一定の複数のフィールドからなる. これらのうち, 分類器でパケットの識別に用いるフィールドをキーと呼び, 各キーが満たすべき条件の記述を**フィルタ**と呼ぶ. キャプチャすべきパケットの条件を記述したフィルタの集合を**フィルタリングポリシ**と呼ぶ.

本稿ではパケット分類器を次のように動作するものとする. すなわち, フィルタリングポリシとパケットが与えられた時, パケットの各キーとフィルタ集合の各フィルタとを照合する. そして, あるパケットに対し, 全てのキーが対応する条件との照合に成功するフィルタが 1 つ以上あればそのパケットをキャプチャする.

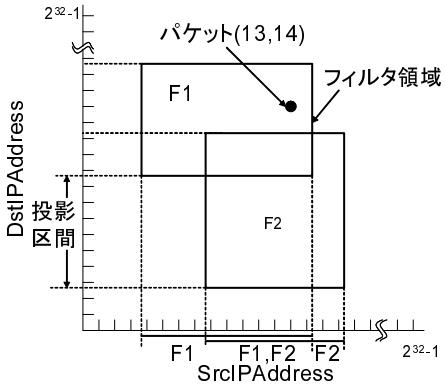

図 1　分類空間の例

3.2　パケット分類器のパケット空間を用いた表現法

パケット分類問題は, 計算幾何学的なアプローチにより多次元空間における点位置決定問題として捉える事ができる [3] [5]. この問題ではまず, キーの数を N とした時, それぞれのキーを軸とする N 次元空間を考える. 全てのパケットをこの N 次元空間の 1 点で表す. このような空間を本稿では**分類空間**と呼ぶ.

例として, 送信元 IP アドレス (以下 $SrcIP$, $0 \leqq SrcIP \leqq 2^{32}-1$) と送信先 IP アドレス (以下 $DstIP$, $0 \leqq DstIP \leqq 2^{32}-1$) をキーとする分類を考える. この分類において次の 2 つのフィルタ $F1, F2$ がある時, 分類空間は図 1 のようになる.

F1:$4 \leqq SrcIP \leqq 14$ かつ $10 \leqq DstIP \leqq 16$

F2:$8 \leqq SrcIP \leqq 16$ かつ $3 \leqq DstIP \leqq 12$

このとき, 図 1 で $F1$, $F2$ の矩形で示されるように, フィルタは分類空間の部分空間を指定することになる. この部分空間を**フィルタ領域**と呼ぶ. フィルタ領域は 1 つ以上の格子点を包含する. 分類空間においてパケットを示す点を含むフィルタ領域が存在するとき, 照合に成功するといい, パケットをキャプチャする. ここでは簡単のために, $SrcIP$ と $DstIP$ の 2 つのみを用いたがキャプチャ条件で使用する全てのキーに対し軸を作成し, パケットがフィルタ領域に包含されるか調べる. 図 1 ではパケット ($SrcIP = 13$, $DstIP = 14$) が点で表されており, $F1$ のフィルタ領域に包含されているので, このパケットは $F1$ との照合に成功し, キャプチャされる. $SrcIP$ の軸では, $F1$ の区間は $[4, 14]$ で $F2$ の区間は $[8, 16]$ となる. $F1$, $F2$ の区間の重なり具合により全区間 $[0, 2^{32}-1]$ を次の 5 区間に分割できる. 重なるフィルタの集合が等しい区間の集合を**等影区間**と呼ぶ. すなわち, $F1$ のみの区間 $[4, 7]$, $F1$, $F2$ の区間 $[8, 14]$, $F2$ のみの区間 $[15, 16]$, 空の区間 $[0, 3]$, $[17, 2^{32}-1]$ の 4 つの等影区間ができる.

3.3　フィルタ領域の近似手法

我々が先に提案した高速パケットキャプチャシステム [8] では分類空間のある軸の各座標と, その座標を含む等影区間を示す識別子との対応表 (分類表) をオンチップのメモリに格納し, パケット到着時にはその表を探索するのみで高速にパケットを分類する. この分類表を少ないメモリ容量で格納するために, フィルタを変更し表のエントリ数を一定数以下に抑える [7].

一般にフィルタを変更すると次の二つの問題が起こる場合がある [11].

問題 1　キャプチャすべきパケットを廃棄してしまう (**false negatives**)

問題 2　廃棄すべきパケットをキャプチャしてしまう (**false positives**)

ファイヤーウォールなどの攻撃パケットの排除のために行われるパケットの分類では全てのパケットに対しフィルタの照合を正確に行う必要がある. しかし, ネットワークの監視・診断システムなどのパケットキャプチャ用のパケット分類では, false positives が起こっても後で診断の際には取り除くことにより, 指定されたパケットを得ることができる. 一方, false negatives が起こると必要なデータがなくなるので, ネットワークに発生したトラブルが検出不可能になる可能性がある. そのため, 文献 [8] で提案したシステムでは false positives は許容するが, false negatives が発生しないことを保障するようにフィルタを変更している. ここで, false positives によるパケットがノイズとなる.

3.3.1　分類条件の近似

分類空間から作られる分類表のエントリ数は等影区間の数に依存する. 近似的空間分割型パケット分類器では等影区間の数が減るようフィルタ領域を拡大することで分類表のデータ量を小さくする.

ある軸で等影区間の組 A と B を 1 つの等影区間にするには, それぞれが持つフィルタ集合を等しくすれば良い. そこで, まず A が持っていて B が持っていないフィルタの集合を求める. このフィルタ集合を B の等影区間に拡大する. 同様に B が持っていて A が持っていないフィルタの集合も A 側に拡大する. このように拡大した各フィルタ領域の拡大部分を**変化領域**と呼び, 2 つの等影区間を 1 つの等影区間とするようにフィルタ領域を拡大する操作を**フィルタ領域の近似**と呼ぶ.

例えば, 図 1 の分類空間は $SrcIP$ 軸では $\{\}, \{F1\}, \{F1, F2\}, \{F2\}$ のフィルタ集合を持つ 4 つの等影区間がある. 図 2 のように $F1$ の領域を右の網掛部に拡大することにより $\{\}, \{F1\}, \{F1, F2\}$ のフィルタ集合を持つ 3 つの等影区間になる. フィルタ領域の近似を一回行うと, 投影区間の数が 1 つ減る. そのため, フィルタ領域の近似を繰り返すことにより等影区間の数を常に一定以下に抑えることができる [7].

ここで, 図 2 の網掛部に新たなフィルタ $F3$ を追加しても $F1$ のフィルタを拡大した場合と同様のパケットが得られる. しかし, この方法では投影区間の持つフィルタ集合が $\{F1\}$ から $\{F1, F3\}$ と変わるだけで, 投影区間の数は変わらない. 従って, エントリ数を削減することができない.

フィルタ領域の近似を行うと, 本来フィルタ領域がない領域にフィルタ領域が拡大されることがある. この領域でキャプチャされるパケットがノイズとなる. 例えば, 図 2 で $SrcIP = 15$ かつ $DstIP = 14$ に相当するパケットは本来破棄されるはずであるが, フィルタ領域の近似によりキャプチャされノイズとなる.

3.3.2　容量に基づく変化領域決定法

ノイズを低減させるためには近似を行う際に変化領域をどのように決めるのかが重要となる. つまり, フィルタリングポリシ

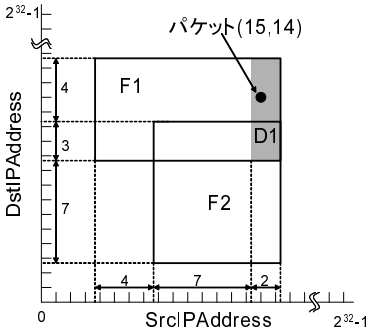

図 2　F1 を右に拡大する近似の例

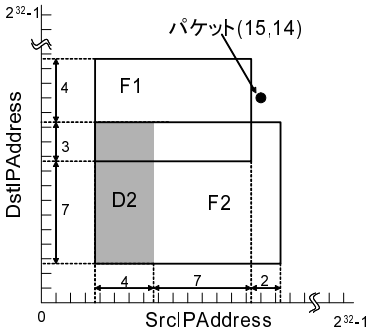

図 3　F2 を左に拡大する近似の例

のどのフィルタをどれだけ変化させるかによってノイズの量が変化する.

容量に基づく変化領域決定法では, 部分空間に含まれる格子点の総数を部分空間の容量と定義する. そして, フィルタリングポリシの各フィルタに対して, 変化させたとするとどのような変化領域になるかを求め, 変化領域の容量の最も小さい変化領域となるフィルタを近似するフィルタと決定する．もし, 全ての格子点にパケットが等確率で出現するならば, 変化領域が小さいほど, ノイズとなるパケットの到着する確率が低くなる. 上記決定法によりノイズを低減することが可能となる.[7]

図 1 の分類空間において SrcIP 軸の投影区間を 1 つ減らす例を示す. この分類空間では, 以下の 2 つ変化領域が考えられる.

D1：F1 を右に拡大したときの変化領域 (図 2 の網掛部)

D2：F2 を左に拡大したときの変化領域 (図 3 の網掛部)

これら変化領域の容量を求める．まず, $D1$ は $15 \leqq SrcIP \leqq 17$ かつ $8 \leqq DstIP \leqq 14$ の領域である．よって $D1$ の容量は $3 \times 7 = 21$ となる．次に, $D2$ は, $3 \leqq SrcIP \leqq 7$ かつ $3 \leqq DstIP \leqq 12$ の領域である．よって変化領域の容量は $5 \times 10 = 50$ となる. この結果, 容量が小さい $D1$ を近似する変化領域とする.

4. 提案システムの実現法

4.1 提案システムの概要

容量に基づく変化領域決定法では全ての格子点においてパケットが等確率で出現するとした. しかし, 実際のネットワークでは格子点によってパケットの出現する確率が異なる. そのため, 変化領域の容量が小さくても大量のノイズがキャプチャされてしまうことがある.

3.3.2 節で挙げた例では図 1 の $F1$, $F2$ のフィルタに対し, $D1$ が選択された. ここで, $SrcIP = 15$ かつ $DstIP = 14$(図 2 の点 $(15, 14)$) のパケットが多数到着した場合を考える. このとき, $D1$ は $D2$ に対し, 変化領域は小さいが多量のノイズがキャプチャされる.

そこで, 提案システムでは事前にキャプチャしたトラヒック特性に基づいてフィルタリングポリシを変更する．ここで, 変更した結果できるフィルタリングポリシを**近似フィルタリングポリシ**と呼ぶ．ノイズを削減するには, フィルタリングポリシのどのフィルタをどのように変化させるかが重要となる.

提案システムでは格子点の重みをその格子点に対するパケットの出現頻度に比例した値として定義する. また, 部分空間に含まれる格子点の重みの総和を部分空間の**重み付き容量**と呼び, 重み付き容量が最小の変化領域を持つフィルタを選択し, 近似する．この計算法を**重み付き容量に基づく変化領域決定法**と呼ぶ.

提案システムは図 4 に示すようにフィルタコンパイラ, パケット分類器, 近似フィルタリングポリシ生成器, ノイズアナライザから構成される. フィルタコンパイラでは第 3 章で述べたパケット分類器のための分類表を作成する. パケット分類器は分類表に従いパケットをキャプチャする. ノイズアナライザはキャプチャしたトラヒックからノイズを分離し分析する. フィルタリングポリシ生成器ではノイズアナライザで分析した, トラヒック特性に基づいて近似フィルタリングポリシを生成する.

4.2 近似フィルタリングポリシ生成器

事前にキャプチャしたトラヒックから書く格子点毎のパケット出現頻度を推定し, パケット出現頻度が高い格子点を変化領域に含まないよう近似するフィルタ領域を決定する．ここで, パケットの格子点毎の出現頻度は次の式で表される単位時間当たりのパケット到着数とする.

$$\text{パケット出現頻度} = \frac{\text{パケット到着数}}{\text{観測時間}} \tag{1}$$

ネットワークの構成やサービスに変化が無ければ, キャプチャされたトラヒックが持つ特性が今後も続くと考えられる. 従って, 過去にパケット出現頻度が高かった格子点は今後もパケット出現頻度が高いと考えられる.

重み付き容量に基づく変化領域決定法では以下の手順により, 変化領域を決定する.

STEP1　フィルタリングポリシの各フィルタに対し, 近似するとどのような変化領域となるかを求める.

STEP2　変化領域に含まれる格子点のパケット出現頻度の合計を求め, 重みつき容量とする.

STEP3　重みつき容量が最小の変化領域を選択する.

ここで, 一度もトラヒックの観測を行っていない変化領域が複数存在した場合, これらの変化領域の重み付き容量は全て 0 となる．このとき, 重み付き容量が 0 の変化領域の中から, 容量に基づく変化領域決定法を用いて変化領域を決定する．変化領域の重み付き容量が小さいことは, その領域に出現するパケットが少ないことを意味する．従って, 重み付き容量に基づく変化領域決定法を用いることにより, ノイズの低減が可能となる.

図 1 の分類空間において $SrcIP$ 軸の投影区間を 1 つ減らす

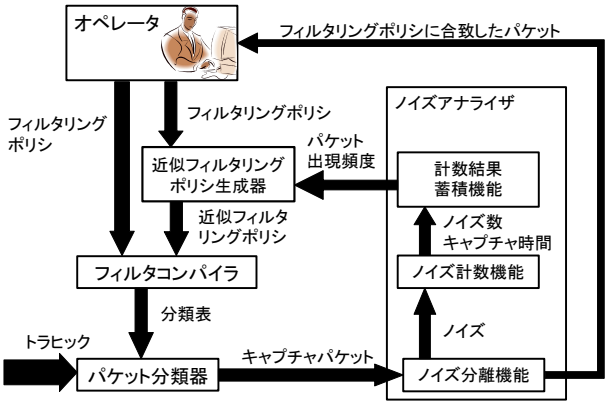

図 4　提案システムの構成図

例を示す．ここで, パケットが $SrcIP = 15$ かつ $DstIP = 14$ の格子点のパケット出現頻度を α, 他の格子点のパケット出現頻度を β とし, $\alpha = 100\beta$ とする．このとき, 次のように変化領域を決定する．

STEP1　$F1$ を近似した場合の変化領域 $D1$(図 2) と F2 を近似した場合の変化領域 $D2$(図 3) の 2 つの変化領域が考えられる．

STEP2　$D1$ の重み付き容量を求める．$D1$ においてフィルタ領域に含まれない格子点数は 8 である．このうち格子点 (15,14) の重みは α で, 他の格子点は β である．したがって, $D1$ の重み付き容量は $\alpha + 8\beta = 108\beta$ となる．次に $D2$ の重み付き容量を求める．$D2$ においてフィルタ領域に含まれない格子点数は 28 で, $D2$ に含まれる格子点の重みは全て β である．したがって, $D2$ の変化領域の重み付き容量は 28β となる．

STEP3　$D1$ と $D2$ の重み付き容量を比較すると $D2$ の重み付き容量の方が小さい．したがって, $D2$ を変化領域とする．

ここではフィルタが 2 つのみの場合を挙げたがフィルタが 3 つ以上ある場合は全てのフィルタの組み合わせについて同様の操作を行い変化領域の重み付き容量が最も少なくなるようフィルタ領域を近似する．

4.3　ノイズアナライザ

ノイズアナライザはノイズ分離機能, ノイズ計数機能, 計数結果蓄積機能から構成される. 以降, 各機能の詳細について述べる．

4.3.1　ノイズ分離機能

近似後の分類表を用いてキャプチャしたパケットには, オペレータが指定したキャプチャ条件に合致したパケットと, ノイズが含まれている. 提案方式ではキャプチャしたパケットをオペレータが与えた近似前のフィルタリングポリシと改めて照合してノイズを抽出する．

4.3.2　ノイズ計数機能

ノイズ計数機能ではキャプチャしたトラヒックから, 各格子点のパケット出現頻度を推定する. 各格子点でパケット出現頻度を求めるためには, 各格子点についてキャプチャを行った観測時間とパケット到着数を記録する必要がある. しかし, 全ての格子点について記録を行うとデータ量が膨大になる. そこで, 提案方式では次の 2 つの方式を用いてデータ量を削減する．

（1） 次元毎のパケット到着数 (軸パケット到着数と呼ぶ) を用いたデータ削減法

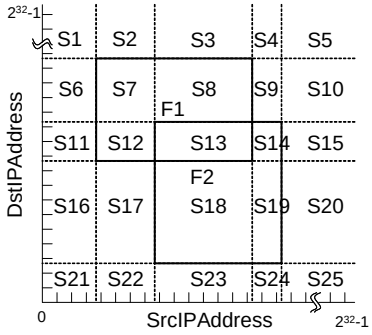

図 5　分類空間をセルに分割した例

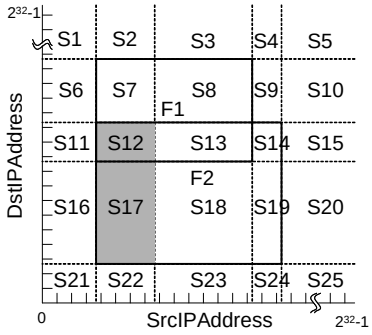

図 6　変化領域に含まれるセルの例

（2） セル毎のパケット到着数を用いたデータ削減法

以降, これら 2 つの方式について述べる．

(1) 軸パケット到着数を用いたデータ削減法 この方式では, パケット到着数を軸毎に独立して記録する. すなわち, $SrcIP = 15$ かつ $DstIP = 14$ の格子点 P にパケットが到着したとき, $SrcIP$ が 15 の軸パケット到着数と $DstIP$ が 14 の軸パケット到着数をそれぞれ 1 増やす. そして, 格子点点 P のパケット到着数を $SrcIP$ が 15 の点の軸パケット到着数と $DstIP$ が 14 の点の軸パケット到着数の積とする．

(2) セル毎のパケット出現頻度を用いたデータ削減法 この方式では投影区間によって分類空間を分割し, 分割された部分空間毎にノイズを保存する. ここで, このように分割された部分空間のことを**セル**と呼ぶ. 3.2 節で用いた図 1 分類空間においてセルは, 図 5 の $S1$〜$S25$ の 25 個の領域となる. フィルタ領域の近似を行う際には, 変化領域は必ず 1 つ以上のセルを組み合わせた領域となる．例えば, 図 5 において $F2$ を左に拡大する近似を考えると, 図 6 変化領域は $S12$, $S17$ を組み合わせた領域となる．そのため, 変化領域に含まれるセルのノイズ出現頻度の合計が変化領域の重み付容量となり, 格子点毎にノイズ出現頻度を保存した場合と同様の重みつき容量を得ることができる．

4.3.3　計数結果蓄積機能

提案方式ではフィルタリングポリシに基づいてキャプチャされたパケットのみを用いてトラヒックの特性を分析するので, パケットの破棄された領域の特性については知ることができない. しかし, 同じネットワークのトラヒックはある程度類似性があると考えられるので, パケットのキャプチャとトラヒックの分析を近似するフィルタ領域を変化させながら何度も繰り返す. そして, 計数結果蓄積機能では過去に計測したノイズの計数結

果と新たに計測したノイズの計数結果を合わせ, パケット出現頻度を計算する.

5. 評 価 実 験

実験により, 次の項目について評価する.

- パケット出現頻度を用いた近似によるノイズ低減効果
- ノイズ出現頻度の保存方式によるメモリ量削減効果

5.1 実験システム

実験のために提案システム (図 4) に基づいて実験システムを作成した. ここで, パケット分類器は文献 [8] に基づいき, ソフトウェアによる, シミュレータとして実現した. また, フィルタコンパイラは文献 [7] に基づいて実現した. 実験システムでは, パケットのヘッダとペイロードを次のように扱う.

（1） パケットの先頭 64 バイトからキーとなるフィールドを選択し, フィルタではペイロードを参照しない.

（2） キーとして選択するフィールドの数を 16 個に固定する.

（3） パケットの分類をハードウェアを用いてテーブル参照のみで高速に行うために, フィールドをバイト単位でキーとして扱うので, 2 バイト以上の範囲で領域照合を指定するフィルタは複数のフィルタに分割して表現する.

（4） フィールドごとに条件指定を許しているので 2 つ以上のフィールドにまたがる条件を持つフィルタは除外する. 例えば, 送信元 IP アドレスと送信先 IP アドレスが同じパケットをキャプチャするというフィルタは使用しない.

上記のような実現のため, サンプルのフィルタに対して次のようにフィルタを変更する.

（1） ペイロードのチェックはせず, フィルタのヘッダに関する記述だけを使用する. 例えば, ”ICMP のエコー要求パケットをキャプチャする”というフィルタは”IP パケットのヘッダの Protocol が ICMP のパケットをキャプチャする”とした.

（2） 使用頻度が高いフィールドから順に 16 バイト分を選択する. 今回選択したキーのフィールドは Protocol, IP destinatoin, IP source, PORT destination, PORT source, LENGTH の 7 つのフィールド, 計 16 バイトである.

（3） ユーザはフィルタリングポリシでフィールド単位でフィルタの条件記述し, フィルタコンパイラにおいて 2 バイト以上の範囲で領域照合を指定するフィルタは分割して複数のフィルタで表現する. 例えば, フィルタリングポリシのフィルタにおいて, ”133.68.13.0 $\leqq SrcIP \leqq$ 133.68.14.255”と指定された場合, フィルタコンパイラは”133.68.13.0 $\leqq SrcIP \leqq$ 133.68.13.255”と”133.68.14.0 $\leqq SrcIP \leqq$ 133.68.14.255”の 2 つのフィルタに分割する.

（4） 2 つ以上のフィールドにまたがるを条件を持つフィルタは本分類器の外部の機能により分類するので, ここでは実験データから削除する.

また, これらによりいくつかのフィルタの内容が等しくなるので, 重複したフィルタは 1 つにまとめる.

5.2 実験用データ

（1） サンプルフィルタリングポリシ

提案方式の効果を確認するためにはサンプルヘッダ内の多くのフィールドを使用しており, かつフィルタ数が多い必要がある. IDS の一種である snort [12] は使用フィルタが公開されており, 上記の条件を満たすので, 実験ではこのフィルタをサンプルとしてフィルタリングポリシを作成する. 今回用いたサンプルでは 2027 個のフィルタがある. これらに対して前述のフィルタの変更を施し, 結果として 454 個のフィルタを得た.

この 454 個のフィルタからランダムに 25 個ずつ抜き出し, サンプルフィルタリングポリシ $P_{25}^1, P_{25}^2, P_{25}^3, P_{25}^4$ を作成した. 同様に 50 個ずつ抜き出し, サンプルフィルタリングポリシ $P_{50}^1, P_{50}^2, P_{50}^3, P_{50}^4$ を, 100 個ずつ抜き出し, サンプルフィルタリングポリシ $P_{100}^1, P_{100}^2, P_{100}^3, P_{100}^4$ をそれぞれ作成した.

（2） サンプルトラヒック

サンプルパケットとして MAWI [13] で公開されているトラヒックアーカイブの中から samplepoint-F の 2006 年 10 月 3 日から 10 月 14 日のものを用いる. このデータは日米間の 100Mbps のリンクで観測された日常的なトラヒックを 14 時から 14 時 15 分の 15 分間分を切り出したものである.

5.3 実 験 項 目

実　験　1

近似したフィルタリングポリシを用いてトラヒックをキャプチャすることによって発生するノイズの量を調べる. 実験ではフィルタリングポリシとして P_{25}^1 を用いる. P_{25}^1 から分類表を作成すると 18 kバイトとなる. これを 10k バイト以下となるよう近似する. また, 近似方式として次の 3 つを用いる.

近似方式 1　容量に基づく変化領域決定法

近似方式 2　重み付き容量に基づく変化領域決定法（軸パケットを用いたデータ削減法を使用）

近似方式 3　重み付き容量に基づく変化領域決定法（セル毎のパケット到着数を用いたデータ削減法を使用）

実験手順を以下に示す.

（1） サンプルトラヒックをキャプチャする

（2） 過去のデータと合わせてキャプチャ結果を分析する

（3） 分析結果に従い近似フィルタリングポリシを生成する

（4） (3) で生成したフィルタリングポリシを使用して翌日のサンプルトラヒックをキャプチャする

（5） (2)〜(4) を繰り返す

このとき, 手順 (1) でキャプチャされたパケットのノイズの量を調べる. ノイズの量を示す値としてノイズ率を次のように定義する.

$$\text{ノイズ率} = \frac{\text{ノイズの数}}{\text{廃棄すべきパケットの数}} \times 100[\%] \qquad (2)$$

ここで, 廃棄すべきパケットの数とはサンプルトラヒックが含むパケットのうち, フィルタリングポリシのフィルタに合致しないパケットの数である. また, ノイズの数とは廃棄すべきパケットのうち, 誤差領域によってキャプチャされるパケットの数である.

実　験　2

P_{25}^1 を分類表が 5k バイト以下となるよう近似し, 実験 1 と同様の手順でノイズ率を求める.

実　験　3

パケット出現頻度の保存に必要なメモリの量を調べる．フィルタリングポリシとして P_{25}^1〜P_{25}^4, P_{50}^1〜P_{50}^4, P_{100}^1〜P_{100}^4 を用いる．また, パケット出現頻度の保存法として次の 3 つの保存方式を用いる．

保存方式 1　格子点毎にパケット到着数を保存
保存方式 2　軸パケット到着数を保存
保存方式 3　セル毎にパケット到着数を保存

これらの保存方式を用いてパケット出現頻度を表として保存し, このとき必要となるエントリ数を調べる．

保存方式 1 と保存方式 2 を用いた場合に必要となるエントリ数は論理的に求めることができる．キーの数を k 個, キーの長さ n bit とすると保存方式 1, 保存方式 2 を用いた場合に必要なエントリ数 E_1, E_2 はそれぞれ以下のようになる．

$$E_1 = 2^{kn} \tag{3}$$

$$E_2 = 2^n k \tag{4}$$

保存方式 3 を用いた場合のエントリ数はサンプルフィルタリングポリシを分析し, 必要なエントリ数を求める．また, 保存方式 3 を用いた場合, フィルタリングポリシのフィルタ領域の境界が全て異なる場合にもっともセルの数が多くなる．このとき, パケット出現頻度の保存に必要なエントリ数 E_3 はキーの数を k 個, キーの長さ n bit, フィルタの数を m すると次のようになる．

$$E_3 = (2m+1)^k \tag{5}$$

5.4　結果と考察

5.4.1　実 験 結 果

実験 1 の結果を図 7 に, 実験 2 の結果を図 8 に示す．横軸はサンプルパケットをキャプチャした日付で縦軸はノイズ率である．実験 3 の結果を図 9 に示す．ここで, 横軸はフィルタの数, 縦軸はエントリ数で, それぞれ対数座標となっている．また, 保存方式 3 の最悪値のとは式 (5) より求めたフィルタリングポリシの全てのフィルタ領域の境界が異なる場合に必要となるエントリ数で, 保存方式 3 の実測値とは実際のフィルタを用いた場合に必要となるエントリ数である．保存方式 3 の実測値はそれぞれのフィルタ数で 4 種類のフィルタリングポリシを用いた場合のそれぞれの値をプロットし, 平均値を線で結んだ．

16 個の 8bit のキーを用いた場合, 保存方式 1 を用いた場合に必要なエントリ数は次のようになる．

$$E = 2^{kn} = 2^{16\times 8} \simeq 3.4 \times 10^{38} \tag{6}$$

同様に保存方式 2 を用いた場合に必要なエントリ数は次のようになる．

$$E = 2^n k = 16 \times 2^8 = 4096 \tag{7}$$

5.4.2　近似手法の比較

図 8 の容量を用いた方式 (近似方式 1) と重み付き容量ををを用いた方式 (近似方式 2, 近似方式 3) のノイズ率を比較すると, 10 月 3 日〜10 日の計測回数が少ない期間ではノイズ率が低い近似方式が日によって異なる．しかし, 10 月 11 日以降では重み付き容量を用いた方式のノイズ率が低くなっていることがわかる．

この原因は次の通り．トラヒックの計測回数が少ない場合, 一度もパケットをキャプチャしたことのない格子点が多数存在する．そのため, フィルタリングポリシを近似する際に, 多くの変化領域の重み付き容量が 0 となる．このような場合, 近似方式 2 や近似方式 3 を用いた場合でも容量の小さい変化領域が選択されるため, 近似方式 1 を用いた場合と変化領域の決定法があまり変わらない．一方, トラヒックの計測回数が多くなると, 多くの変化領域でパケット出現頻度が推定可能となる．そのため,

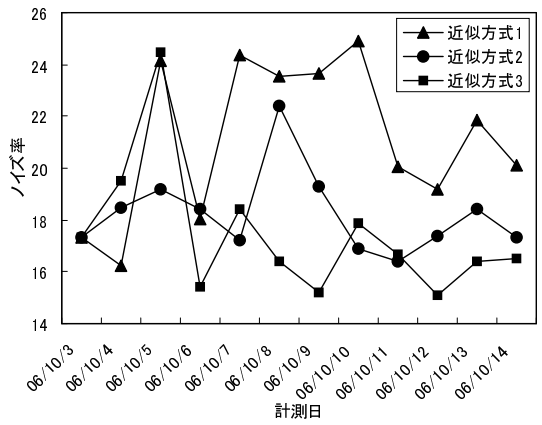

図 7　25 個のフィルタを 10 k バイトに近似した場合のノイズ率

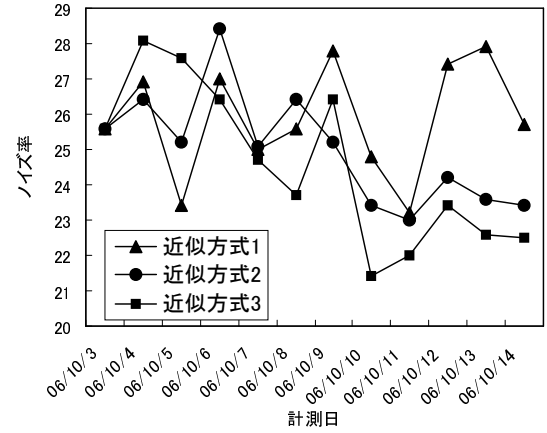

図 8　25 個のフィルタを 5 k バイトに近似した場合のノイズ率

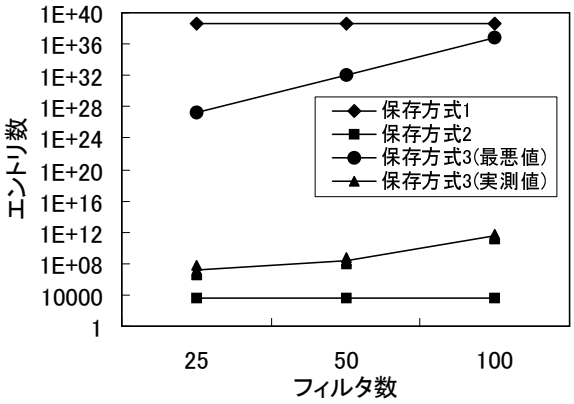

図 9　パケット出現頻度の保存に必要なエントリ数

パケット出現頻度の高い変化領域を含まない近似が可能となり, 重み付き容量を用いた方式を用いるとノイズ率が減少する.

図 8 の近似方式 2 と近似方式 3 を比較すると, 計測回数が増えると近似方式 3 のほうがノイズ率が低く, 最終的に 5%程度ノイズ率が低くなるなっている.

この原因は次の通り. トラヒックの計測回数が少ない場合, 先に述べたように近似方式による変化領域の決定法があまり変わらない. 一方, 計測回数が多くなると, 重み付き容量によって変化領域が決定されるようになる. 近似方式 2 ではパケット出現頻度を軸に射影して保存するので, 変化領域の重み付き容量に近似する変化領域に含まれない領域で発生するパケット出現頻度の影響が入ってしまう. 一方, 近似方式 3 では変化領域のパケット出現頻度が正確に保存できる. 以上より, 近似方式 2 では, 近似方式 3 と比べトラヒック特性の推定精度が落ちるので, 近似方式 3 のほうがノイズ率が低くなる.

5.4.3 パケット出現頻度の保存法の比較

図 9 の保存方式 3 の実測値をみるとフィルタの数が同じならば, フィルタリングポリシによってパケット出現頻度の保存に必要なエントリ数は大きく変わらないことがわかる.

図 9 の保存方式 1 を見ると, 格子点毎に保存すると 10^{38} 個のエントリが必要となることがわかる. 実際に 10^{38} 個ものデータを保存するのは不可能である. そのため, パケット出現頻度の保存に必要なメモリ量を削減することが必要であるといえる.

図 9 の保存方式 3 の実測値をみると, フィルタ数が 25 個の場合は 5×10^6 程度の少量のエントリですむことがわかる. また, フィルタ数 100 個の時には最悪の場合 7×10^{34} 個のエントリが必要となるが, 実際には 2×10^{11} 個のエントリですむことがわかる. しかし, 全てのエントリに 1 バイトのデータを入れた場合, 全てのデータを保存するのに数 T バイトのメモリが必要となり現実的ではない. ここで, 実際のトラヒックではパケットが現れるセルと現れないセルがある. そのため, 全てのセルに対してエントリを作成するスパースな表となってしまう. そこで, パケットの発生したセルに対してだけエントリを作成し, パケット出現頻度を保存することで必要なメモリ量を低減可能であると考えられる.

以上のことから, フィルタ数が 25 個の場合には重み付き容量に基づく変化領域決定法（セル毎のパケット到着数を用いたデータ削減法を使用）が適しているといえる.

また, フィルタ数が 100 個の場合には, セル毎にパケット到着数を保存する方式は実現上問題があるが, 軸パケット到着数を用いた方式は少量のエントリで保存可能である. 図 8 を見ると, 軸パケット到着数を用いた方式 (近似方式 2) はセル毎にパケット到着数を保存する方式 (近似方式 3) と比べノイズが増加している. しかし, 従来の方式 (近似方式 1) と比べると十分にノイズが減少している. そのため, 重み付き容量に基づく変化領域決定法（軸パケットを用いたデータ削減法を使用）が適しているといえる.

6. おわりに

本稿では, フィルタリングポリシをトラヒック特性に適応させる機能を有するパケットキャプチャシステムを提案した. 実験により, 提案システムを用いることで, ノイズを低減可能であることを確認した.

今後の課題として次のものが挙げられる.

検証実験において snort のフィルタから生成したフィルタリングポリシを用いたが, 今後, 異なるフィルタリングポリシを用いて実験を行い, どのようなフィルタリングポリシでどのような効果が出るかを明確にする.

実験システムでは, パケット出現頻度を保存するためのエントリをあらかじめ全て用意している. そのため, フィルタ数が多い場合ではセル毎に保存することができない. ハッシュ等を用いて, パケットの出現した要素のエントリだけを作成することで, 必要なメモリ量を削減する必要がある.

現状では 2 つ以上のフィールドにまたがる条件を用いることができないので, これを可能とする必要がある.

謝　辞

日頃, 御指導ならびに御助言をいただいた, 野呂昌満教授に心から深く感謝致します. また, 本研究の一部は平成 18 年度科学研究補助金 (特定領域研究 18049038 及び基盤研究 (C)18500050) の研究助成によるものである.

文　献

[1] Anthony J. McAuley and Paul Francis, “Fast routing table lookup using CAMs, ” INFOCOM, Vol.3, pp.1382–1391, 1993.

[2] V.Srinivasan, Subhash Suri, and George Varghese, “Packet classificationusing tuple space search, ” SIGCOMM, pp.135–146, September 1999.

[3] T.V.Lakshman and Dimitrios Stiliadis, “High-speed policybased packet forwarding using efficient multi-dimensional range matching”, SIGCOMM, pp.203-214, 1998.

[4] Pankaj Gupta and Nick McKeown, “Packet classification on multiple fields, ” SIGCOMM, pp. 147–160, August 1999.

[5] N.Takahashi, “A systolic sieve array for real-time packet classification, ” Journal of IPSJ, Vol.42, No.2, pp.146-166, 2001.

[6] 高橋直久, “フィルタ sieve 関数の部分計算に基づく実時間パケット分類器, ” 日本ソフトウェア科学会インターネット技術ワークショップ WIT 99, pp. 190–197, 1999.

[7] 大須賀怜, 片山喜章, 高橋直久, “近似機能を有する空間分割型パケット分類器, ” 情報処理学会論文誌, Vol.47, No.4, pp.1195-1208, April 2006.

[8] 加藤和夫, 大須賀怜, 高木淳史, 片山喜章, 高橋直久, “スケーラブルパケットキャプチャシステムの実現, ” 日本ソフトウェア科学会インターネット技術ワークショップ WIT 2003, pp.32-39, November 2003.

[9] 清水奨, 風間一洋, 廣津登志夫, 後藤滋樹, “リアルタイム圧縮によるパケットキャプチャの高速化”, 情報処理学会論文誌, Vok.47, No.SIG 7(ACS 14), pp.183–193, May 2006.

[10] Cisco Systems: Netflow, `http://www.cisco.com/warp/public/732/Tech/netflow`.

[11] Stephen Northcutt, Lenny Zeltser, Scott Winters, Karen Kent, Ronald W.Ritchey, “Inside Network Perimeter Security Second Editoin, ” SAMS Publishing, pp.205–206, Mar 2005.

[12] Sourcefire: The open source network intrusion detection system, `http://www.snort.org/`.

[13] WIDE Project: Mawi working group traffic archive, `http://tracer.csl.sony.co.jp/mawi/`.

[14] 中村陸, 片山喜章, 高橋直久, “トラヒック特性に基づく近似機能を有する空間分割型パケットキャプチャシステム, ” 信学技報, vol.106, no.151, IN2006-48, pp.79-84, July 2006.